Service Adaptation Using Fuzzy Theory in Context-aware Mobile Computing Middleware*

Jiannong Cao[1], Na Xing[1,2], Alvin T.S Chan[1], Yulin Feng[2] , Beihong Jin[2]
[1]*Internet and Mobile Computing Lab*
Dept. of Computing, Hong Kong Polytechnic University
{csjcao, cstschan}@comp.polyu.edu.hk
[2]*Technology Center of Software Engineering*
Institute of Software，Chinese Academy of Sciences
{nxing, feng, jbh}@otcaix.iscas.ac.cn

Abstract

*Context-aware mobile computing middleware is designed to automatically adapt its behavior to changing environment. To achieve this, an important issue to be addressed is how to effectively select services for adaptation according to the user's current context. Existing work does not adequately address this issue. In this paper, we propose a **F**uzzy-based **S**ervice **A**daptation **M**odel (**FSAM**) that can be used in context-aware middleware. We formulate the service adaptation process by using fuzzy linguistic variables and membership degrees to define the context situations and the rules for adopting the policies of implementing a service. We propose three fitness functions to calculate the fitness degree for each policy based on the distance of fuzzy status between the policy and the current context situation. The decision for service adaptation is achieved by selecting the policy with the largest fitness degree. A context-aware application scenario called Campus Assistant is used to exemplify the proposed service adaptation process and demonstrate its effectiveness.*

1. Introduction

Mobile computing imposes new challenges in designing computer hardware and software due to user mobility, the diverse types of devices used, resource constraints, and the dynamic nature in execution context. *Context-aware mobile computing middleware* [1,2] provides abstraction and support for application programmers to ease the task of developing mobile applications, ensuring acceptable QoS and allowing for adaptation to changes in the operating environment. An important issue to address in designing a context-aware middleware is how to effectively select services for adaptation according to the user's current context. However, this issue has not been adequately addressed in existing work which has been focused either on the software realization of services configuration or on a specific scenario or domain [2, 8].

This paper is concerned with the formulization and development of a Service Adaptation Model for context-aware mobile computing middleware. We propose the design of a *Fuzzy-based Service Adaptation Model* ***(FSAM)*** using fuzzy theory [9]. As we will show in this paper, context information is largely uncertain and vague, and introducing fuzzy theory into the adaptation process would make the process more flexible and adaptive.

The remainder of the paper is structured as follows. Section 2 describes related work. Section 3 presents an example scenario to illustrate the service adaptation problem. Section 4 describes the design of the fuzzy-based service adaptation model. Section 5 illustrates how the model works with the example application and evaluates its effectiveness. Section 6 concludes the paper and discusses our future work.

2. Related Work

To provide support for building high performance mobile computing applications, research in the field of middleware systems has proliferated in the recent years [10]. Most context-aware middleware uses *reflective* architecture with mechanisms for the dynamic deployment and re-configuration of the underlying services [1,2]. In this section, we briefly describe

* This work is supported by Hong Kong Polytechnic Univ. under HK PolyU Research Grants G-YD63 and 4-Z073, and National Grand Fundamental Research Program of China under Grant No.2002CB312005, the Chinese National "863" High-Tech Program under Grant No.2004AA112010.

Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'05)
1533-2306/05 $20.00 © 2005 **IEEE**

existing works on context-aware middleware and system related to service adaptation.

The CARISMA project [2] is a context-aware middleware system made up from adaptable services. It uses context-aware application-specific semantic information in an "Application Profile" encoded in XML. When used, the middleware checks the application profile document and compares with the current execution context to evaluate which behavior or policy the service component should use. CARISMA proposed a microeconomic approach that relies on a particular type of sealed-bid auction to resolve the service policy confliction at execution time.

Systems have also been developed to directly address adaptation issues. The Chisel system [3] introduced a dynamic services adaptation framework which decomposes the particular aspects of a service object into multiple possible behaviors. Whenever the context information changes, the service object will be adapted to use the different behaviors according to the adaptation policy. The Functionality Adaptation method in [4] describes proxy-based context-aware adaptation of service code modules, by which the functionality of a service is adapted based on the estimation of the resource usage required for the execution. Policy-driven Mobile Agents [5] and Case-Based Situation Assessment [6] are also proposed for adaptation in context-ware system.

Compared with the above works, our service adaptation model aims at formulating and designing a policy selection mechanism, and we employ the fuzzy theory to solve the general problem of developing a generic service adaptation model. Existing works either work on a specific problem or focus on the software realization of services and policy rules.

Also relevant to our work is the research on using fuzzy-based method for control and adaptation. A survey of techniques for using fuzzy theory to adapt QoS requirements in communication networks can be found in [7]. An example as described in [8] uses fuzzy control theory for QoS adaptation in distributed multimedia applications. The QoS-related approaches are usually developed for a specific domain (e.g., multimedia applications), and the related QoS parameters just constitute a subset of context information (e.g., bandwidth, CPU). Again, they are not targeted at a generic service adaptation model.

3. An Example

Before we proceed to describe the proposed FSAM model, in this section we first illustrate the service adaptation problem by describing a hypothetic example application. The example will also be used in Section 5 to illustrate how the FSAM approach can be applied.

Let us imagine a university student, Alice, roaming around campus, and using a PDA installed with a context-aware *Campus Assistant* application. We will assume that the application runs two services *Chat Service* and *Email Service*. Alice uses the PDA to send and receive email through the school's email server, and to chat with classmates.

The services provided by the Campus Assistant application have alternative policies according to the real-time context. For example, in order to distribute messages among chatting participants, FSAM helps applications make a suitable choice from among three policies: *textChat*, which allows the delivery of text messages; *voiceChat*, which allows the exchange of voice messages; and *videoChat*, which allows the exchange of video messages. The email service operates in a similar fashion. Students can check their emails by using one of the following five policies: *headMail*: which delivers only the mail header and a sign indicating that the receiving process is not finished and will continue when a higher network bandwidth and other resource are available; *fullMail*: which delivers mail in full-text; *encryptedMail*: which sends encrypted mail; *bigMail*: which delivers mail in full-text and attachments; and *encryptedBigMail*, which delivers encrypted big-mails.

Based on the modeling and assessment of the current contextual information of Alice, the policies of a service should be automatically selected to adapt the service to the prevailing conditions.

4. The FSAM Framework

As shown in Figure1, a context-aware middleware is composed of five parts. These parts collaborate with each other to fulfill middleware function. To monitor the contextual changes, *Context Manager* checks the sensor network as well as the user profile regularly, and then abstracts the context information into a FSAM-practicable form. *Resource Manager* inspects the system resource usage and informs FSAM. When a service adaptation decision is made by FSAM, *Configuration Manager* will be triggered to re-configure the *Service Space*, which is the organization of all the on-the-job services. *Configuration Manager* maintains the metadata of all the alternative service implementations, and provides the metadata as the rule for service selection process.

As the core component, FSAM takes as input the context information, service policy definition, and intervention from users and system. These are obtained from other components such as context manager, resource manager and configuration manager. The output of FSAM is the adaptation decision which will be used by the configuration manger for re-configuring the services. In this paper, we focus on the adaptation process used by FSAM, called *Fuzzy-based Service Adaptation Process (FSAP)*.

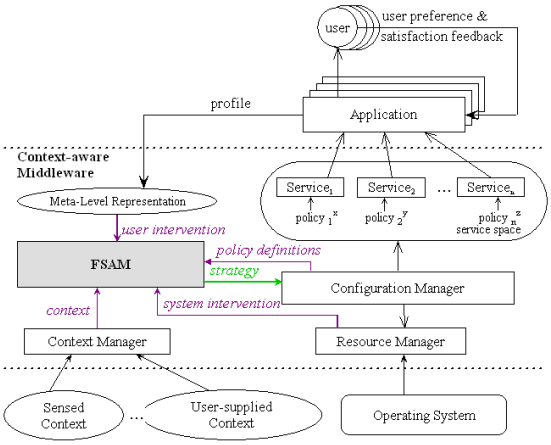

Figure1. Reference Architecture of Context-aware Middleware

4.1. The Fuzzy Service Adaptation Process

The traditional fuzzy control process [9], as shown in Figure2, consists of three stages: *fuzzification*, *reasoning by inference engine*, and *defuzzification*. During fuzzification, predefined membership functions for each linguistic variable are used to determine the degree of membership of a crisp value for the linguistic variable. Then, the inference engine refers to the predefined fuzzy rule base to derive the linguistic values for the intermediate and output linguistic variables. Once the output linguistic values are available, in defuzzification, the final crisp values are produced from the output linguistic values.

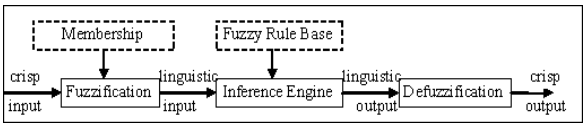

Figure2. Fuzzy Control Process

The proposed FSAP is similar to the traditional fuzzy control process in that it is in the following three steps:

Step_1: Fuzzification. Each context information is represented by a linguistic variable, which may be associated with several linguistic values. Each linguistic value is represented by a predefined application-related membership function (e.g. $\mu_{\text{Network\_maxRate high}}(x)$ and $\mu_{\text{Network\_maxRate low}}(x)$, where x is the crisp value of the corresponding linguistic variable). By using the membership functions we translate the input context crisp value into a set of pairs, each consisting of a linguistic value and a membership degree for that value. The fuzzified context information is then combined into a fuzzified context situation (see definition in Section 4.2).

Step_2: Calculation of fitness degree for each policy. In context-aware mobile middleware, a service can be delivered using several policies and each policy is associated with a particular context situation. We assume that a service can be delivered using only one policy at any time. During the inference process in FSAP, each policy will be assigned a fitness degree indicating to what degree the policy is suitable for being used under the current context situation. The fitness degree is assigned by using a fitness function, which calculates the fuzzy distance between the policy's most suitable context situation and the current context situation. The fitness degree will decrease as the fuzzy distance increases. The most suitable context situations and some additional intervention rules (e.g. application profile, user preference and system feedback) form the rule base of FSAP.

Step_3: Adaptation. For a requested service, the policy with the largest fitness degree will be selected as the best policy for delivering the service under the current context situation.

4.2. Formulized Solution

To introduce the proposed formulized solution, we first give the definition of the concepts and terminologies used in FASP, and then design fitness functions base on the concepts.

Definition 1 (Service): A service represents a function that is provided by the middleware and invoked by a mobile application. Let $S=\{s_1, s_2, s_3, \ldots, s_q\}$ be the set of services provided by the middleware, where, s_i $(1 \le i \le q)$ represents the i-th service, q is the number of services. We use S_{need} to denote the set of services requested by the mobile application.

Definition 2 (Policy): A policy represents a method used to deliver a service with a certain resource requirement and quality-of-service condition. Let $P_i = \{p_i^1, p_i^2, \ldots, p_i^{mi} \mid i \in [1, q]\}$ be a set of policies which can be adopted for delivering the i-th service s_i ($s_i \in S$), where, p_i^j $(1 \le j \le m_i)$ represents the j-th policy of s_i, m_i is the number of all policies for s_i.

Definition 3 (Context): Let $C = \{c_1, c_2, \ldots, c_n\}$ be a set of context which are monitored by the middleware, where, c_i $(1 \le i \le n)$ represents the i-th context information, k is the number of all monitored context.

Definition 4 (Context Situation): A context situation is a combination of context information. Let $LV = \{lv_1, lv_2, \ldots, lv_k\}$ be a set of linguistic values. The context situation at time t is denoted by SI(t) and represented by a set of 3-element tuples:

$$SI(t)=\{(c_a, lv_b, \mu_{c_a\, lv_b}(\text{value\_of}(c_a,t)) \mid c_a \in C, a \in [1,n], lv_b \in LV, b \in [1,k]\} \quad (1)$$

where, $c_a (1 \le a \le n)$ is a certain context information (eg. c_1 = Network_maxRate); $lv_b (1 \le b \le k)$ is a linguistic value (eg. lv_2 = high); $value\_of(c_a,t)$ represents the value of context c_a at time t; $\mu_{c_a\, lv_b}(x) \in [0,1]$ is the

pre-defined membership function of "c_a is lv_b", which indicates when c_a = x, to what degree c_a is lv_b, (eg, if c_1 = Network_maxRate, lv_2 = high, t=current, value_of(c_1, t) = value_of(Network_maxRate,current) = 10M bps, then a possible value of $\mu_{c_1 lv_2}$ (value_of(c_1,t)) can be $\mu_{c_1 lv_2}$ (value_of(c_1,t)) =$\mu_{\text{Network\_maxRate high}}$(10M bps) = 0.5, the value 0.5 means that Network_maxRate is high with degree of 0.5, when Network_maxRate is 10M bps.)

Given a service s_i ($s_i \in S_{need}$), each policy p_i^j in P_i is associated with a context situation that is most suitable for p_i^j. Here, by "most suitable" we mean the best balance of the tradeoff between resource consumption and QoS. Such a best-suitable context situation is referred to as the *Standard Reference* for p_i^j (denoted by $SR(p_i^j)$). If the actual context situation is better than $SR(p_i^j)$, (e.g. actual Network_maxRate is higher than the defined value in $SR(p_i^j)$), then there is waste of resource if p_i^j is used to deliver the service; similarly if the actual context situation is worse than $SR(p_i^j)$, using p_i^j will not obtain the expected QoS.

Definition 5 ($SR(p_i^j)$): Given a set of linguistic values LV = {lv_1, lv_2,…, lv_k}, $SR(p_i^j)$ can be represented by a set of 3-element tuples:

$$SR(p_i^j)=\{(c_a, lv_b, \mu_{c_a lv_b}(\text{best\_value\_of}(c_a)) \mid c_a \in C, a \in [1,n], lv_b \in LV, b \in [1,k]\} \quad (2)$$

Equation (2) is almost the same as equation (1) except that the function *value_of* (c_a, *t*) is replaced by *best_value_of*(c_a), which represents the most suitable value of context c_a when we use policy p_i^j to deliver service s_i. For a given set P_i, we call the aggregation {$SR(p_i^1)$, $SR(p_i^2)$, … , $SR(p_i^{mi})$}, $p_i^j \in P_i$, as the *Standard Reference Depositary* of P_i and denote it by $SRD(P_i)$.

Definition 6 (FSAP): A FSAP is a mapping process from the current context situation *SI(current)* to a set of suitable policies $P_{suitable}$, where each element of $P_{suitable}$ is the most suitable policy for a certain service $s_i \in S_{need}$, The number of elements in $P_{suitable}$ is equal to the number of elements in S_{need}.

Now we are ready to define the fitness functions. Although the aim of FSAP is to obtain $P_{suitable}$, the selection processes for elements in $P_{suitable}$ are similar and irrelevant from each other, thus the key point of FSAP is how to select the most suitable policy for a given service with making the best use of current resource and enhancing the user's satisfaction. In practice, a context situation may not be exactly matched with any $SR(p_i^j)$. In order to select the most suitable policy from P_i, we should use proper fitness function to evaluate all the policies in P_i, so as to make the best choice.

Definition 7 (Fitness Function): Let $FD(p_i^j)$ be the fitness degree for policy p_i^j under current context situation. Given a service $s_i \in S_{need}$, the Fitness Function (FF): $SI(current) \times SR(p_i^j) \rightarrow FD(p_i^j)$, is a mapping from the current context situation and Standard Reference p_i^j to the fitness degree of policy p_i^j.

Here, we propose three different fitness functions:

Function_1:$FF(SI(current),SR(p_i^j))=$

$$\frac{1}{\sum_{i=1}^{size\_of(SR(p_i^j))} \left|\mu(best\_value\_of(c_i)) - \mu(value\_of(c_i, current))\right|}$$

Function_2:$FF(SI(current),SR(p_i^j))=$

$$\frac{1}{\sum_{i=1}^{size\_of(SR(p_i^j))} (\mu(best\_value\_of(c_i)) - \mu(value\_of(c_i, current)))^2}$$

Function_3:$FF(SI(current),SR(p_i^j))=$

$$\frac{1}{\sum_{i=1}^{size\_of(SR(p_i^j))} \left|\mu(best\_value\_of(c_i)) - \mu(value\_of(c_i, current))\right|^{l_i}}$$

where, $size\_of(SR(p_i^j))$ represents the number of tuples in $SR(p_i^j)$, $\mu(x)$ is the membership function appears in i-th vector, l_i is a natural number.

The concept of fitness function is inspired by the membership function in classical fuzzy logic theory. But the two are different, in that the value of fitness degree is a positive number but not limited into [0,1], and the sum of fitness degree for all the policies for one service is not 1. The value of fitness degree only indicates that to what degree one policy is suitable for the current environment. In the above three functions, the denominators are for the calculation of the distance between *SI(current)* and $SP(p_i^j)$. After obtaining the fuzzy distance, we calculate its reciprocal to get the fitness degree, i.e. when the distance between SI(current) and $SR(p_i^j)$ is 0, which means *SI(current)* and $SP(p_i^j)$ are completely consistent, then the fitness degree is infinity. Otherwise, the fitness degree will decrease with the increment of the distance.

5. Example and Discussion

In this section, we use the Campus Assistant application example to illustrate how FSAP can be used and evaluate the effectiveness of three fitness functions.

5.1. Service Adaptation in Campus Assistant

We use Function_3, and let all *ls* in Function_3 take the value of 2 for chat service. For the email service, we let *ls* for Network_maxRate and CPU_clockRate take the value of 2 and *l* for RAM_freeSpace take the value of 3, i.e., we assume that Network_maxRate and CPU_clockRate are more important than RAM_freeSpace in making the decision for service adaptation.

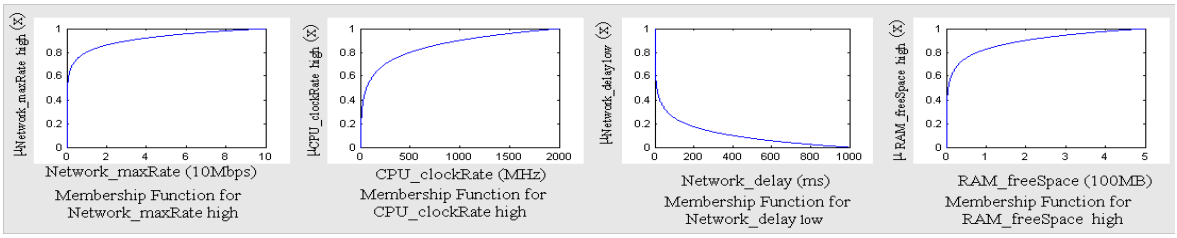

Figure3. Predefined Membership Functions

We assume that before carrying out service adaptation the middleware has obtained the following information:

- S = {chat, email};
- P_1={textChat(p_1^1), voiceChat(p_1^2), videoChat(p_1^3)};
- P_2={headMail(p_2^1),fullMail(p_2^2),encryptedMail(p_2^3), bigMail(p_2^4), encryptedBigMail(p_2^5)};
- **C**={Network_maxRate(c_1),CPU_clockRate(c_2), Network_delay(c_3),RAM_freeSpace(c_4)};
- LV = {high, low}
- SI**(t)**={(Network_maxRate, high, $\mu_{\text{Network\_maxRate high}}$(value_of(Network_maxRate, t))), (CPU_clockRate, high, $\mu_{\text{CPU\_clockRate high}}$(value_of (CPU_clockRate, t))),(Network_delay, low, $\mu_{\text{Network\_delay low}}$(value_of (Network_delay, t))),(RAM_freeSpace, high, $\mu_{\text{RAM\_freeSpace high}}$(value_of (RAM_freeSpace, t)))}

where, the predefined membership functions given by a certain application are shown in Figure3**.**

- User Privacy Intervention **(I_{UPI})**: If user want to protect its privacy then P_1' = {p_1^1, p_1^2} and P_2' = { p_2^1, p_2^3, p_2^5}, else P_1' = P_1 and P_2' = P_2.
- Application Power Intervention (I_{API}): If Power_batteryLevel is less than 5%, then P_1' = P_1 and P_2' = {p_2^1}.
- The most suitable context values for policies in P_1 and P_2 as shown by Table 1.

Table1. Most Suitable Context Values for Policies

	C_1(kbps)	C_2(MHz)	C_3(ms)	C_4(MB)
p_1^1	4	20	500	0.2
p_1^2	200	300	10	4
p_1^3	10000	1000	0.2	200
p_2^1	2	4	---	0.2
p_2^2	10	10	---	0.4
p_2^3	10	100	---	10
p_2^4	500	50	---	2
p_2^5	500	1000	---	100

Given the current context as shown in Figure4, the FSAP performs the following steps for achieving service adaptation:

Network_maxRate = 10kbps CPU_clockRate = 10MHz Network_delay = 100ms

RAM_freeSpace = 10MB I_{UPI} *= 0* *Power_batteryLevel = 40%*

Figure4. Current Context Information

Step_1: Use the predefined membership functions to translates current context information into SI(current) = {(Network_maxRate, high, 0.20), (CPU_clockRate, high, 0.23),(Network_delay, low, 0.25), (RAM_freeSpace, high, 0.58)}.

Step_2: Use I_{UPI} and I_{API} to filter P_1 and P_2, and get P_1' = P_1 and P_2' = P_2.

Step_3: According to the values given in Table 1, use predefined membership functions to calculate SRD(P_1') and SRD(P_2') shown in Table 2.

Table 2 SRD(P_1') and SRD(P_2')

		C_1 High	C_2 High	C_3 Low	C_4 High
SRD(P'$_1$)	SR(p_1^1)	0.12	0.33	0.08	0.15
	SR(p_1^2)	0.46	0.72	0.50	0.48
	SR(p_1^3)	0.80	0.90	0.92	0.90
SRD(P'$_2$)	SR(p_2^1)	0.06	0.10	---	0.15
	SR(p_2^2)	0.20	0.23	---	0.23
	SR(p_2^3)	0.20	0.57	---	0.58
	SR(p_2^4)	0.54	0.47	---	0.40
	SR(p_2^5)	0.54	0.90	---	0.83

Setp_4: Use Function_3 to calculate fitness degree for each policy and get the results shown in Figure5.

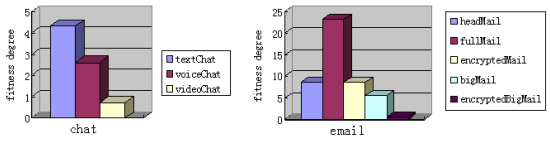

Figure5. Fitness Degree for Policies

Step_5: According to the fitness degrees of policies, derive the set $P_{suitable}$ = { *textChat ,fullMai, }* as the current adaptation strategy.

5.2. *l* Value Discussion

In this section, we will show how the service adaptation decision is effected by the *l* value. The same example is used, except that during the adaptation process in Step 4 all the three fitness functions are used to respectively calculate the fitness

degrees of the policies for email service. The fitness degrees are shown in Figure 6.

Table3. Comparison between Current Context and Best Values for Two Policies

	C_1(kbps)	C_2(MHz)	C_4(MB)
p_2^2	10	10	0.4
p_2^3	10	100	10
current	10	10	10

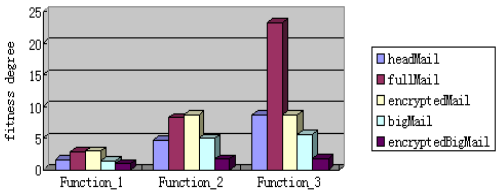

Figure6. Fitness Degrees for EmailChecker Policies Calculate by Three Functions

From Figure6, we can see that by using Function_1 and Function_2 encryptedMail is selected as the most suitable policy, but Function_3 tells us fullMail is the best. As listed in Table3 the current context values for Network_maxRate and CPU_clockRate are consistent with the best values of fullMail; and the current context value for RAM_freeSpace is consistent with the best value for encryptedMail. Since we have assumed in section 5.1 that RAM_freeSpace is less important than the other two contexts for the final choice, the most suitable policy here should be fullMail, which is same as the choice made by Function_3. So, we get the conclusion that properly assigning *l* values will make Function_3 much better to response to the current context. In current FASM, the *l* values are predefined in the application profile base on the developer's experience. Actually, a systematic solution for automatically choosing the right *l* values will lead to a non-linear optimization problem, which may be discussed in our future paper.

6. Conclusion and future work

In this paper, we addressed an important problem of designing context-aware mobile computing middleware, namely service adaptation control. We presented the design of a fuzzy-based service adaptation model, called FSAM, for context-aware mobile middleware. FSAM gears its adaptation based on the changing contextual information and the requirements of users, applications and execution environment. FSAM is the core part of a context-aware middleware, which is designed towards providing strategies to implementing a service re-configuration mechanism. We focused on the development of the fuzzy service adaptation process used in FSAM. By formulating the service adaptation process and introducing the fitness functions we proposed a flexible mathematical method to assign fitness degrees for service policies.

Currently, besides finding systematic manner to optimize the *l* weight value, we are investigating on factors that affect the adaptation quality and performing further evaluations of the context adaptation methods. As future work, we will implement a prototype of the proposed framework. We will also investigate efficient methods for predicting context changes according to the current settings and user mobility patterns to achieve pro-active service adaptation.

References

[1] Chan,A.T.S.; Siu-Nam Chuang; "MobiPADs: A Reflective Middleware for Context-Aware Mobile Computing"; IEEE Transactions on Software Engineering, Volume: 29, Issue: 12 pp.1072 – 1085, Dec 2003

[2] Capra, L.; Emmerich,W.; Mascolo,C.;"CARISMA: context-aware reflective middleware system for mobile applications"; IEEE Transactions on Software Engineering, Volume:29, Issue:10, pp.929 – 945, Oct. 2003

[3] Keeney, J.; Cahill, V. "Chisel: a policy-driven, context-aware, dynamic adaptation framework"; Proceedings of IEEE 4th International Workshop on Policies for Distributed Systems and Networks, pp.3 – 14, 4-6 June 2003

[4] Vivien Wai-Man Kwan; Francis Chi-Moon Lau; Cho-Li Wang; "Functionality adaptation: a context-aware service code adaptation for pervasive computing environments" Proceedings of IEEE/WIC International Conference on Web Intelligence, pp. 358 – 364, 13-17 Oct. 2003

[5] Kun Yang; Alex Galis; Chris Todd, "Policy-driven Mobile Agents for Context-aware Service in Next Generation Networks" MATA 2003- IFIP 5th International Conference on Mobile Agents for Telecommunications, 8-10.10.2003

[6] Anders Kofod-Petersen, Angar Aamodt, "Case-Based Situation Assessment in a Mobile Context-Aware System" Proceedings of Workshop Artificial Intelligence in Mobile System 2003(AIMS2003)at Ubicomp 2003,October 12

[7] Koliver, C., Farines, J.-M., and Nahrstedt, K. O. "QoS Adaptation Based on Fuzzy Theory" Soft Computing for Communications, L. Wang, Ed. Springer-Verlag, pp. 245–267, 2004

[8] Baochun Li; Nahrstedt, K.;"A control-based middleware framework for quality-of-service adaptations" IEEE Journal on Selected Areas in Communications, Volume: 17 , Issue: 9 pp. 1632 – 1650 , Sept. 1999

[9] Constantin Von Altrock; Fuzzy Logic and Neuro-Fuzzy Applications Explained ", Prentice Hall, 1995

[10] Capra, L.; "Mobile Computing Middleware for Context-Aware Applications" Proceedings of the 24th International Conference on Software Engineering, pp.723–724,2002